NATIONAL INSTITUTE OF TECHNOLOGY

KARNATAKA, SURATHKAL

EC-386

MINI PROJECT REPORT

SUBMITTED BY: AKASH -A, ROHAN -J,NAVRATAN

ROLL NO's: 191EC102,191EC147,191EC133

TOPIC: CROPLAND CLASSIFICATION USING OPTICAL AND RADAR DATA

PROJECT MENTOR: RAGHAVENDRA B.S SIR

CROPLAND CLASSIFICATION USING OPTICAL AND RADAR DATA

INTRODUCTION:

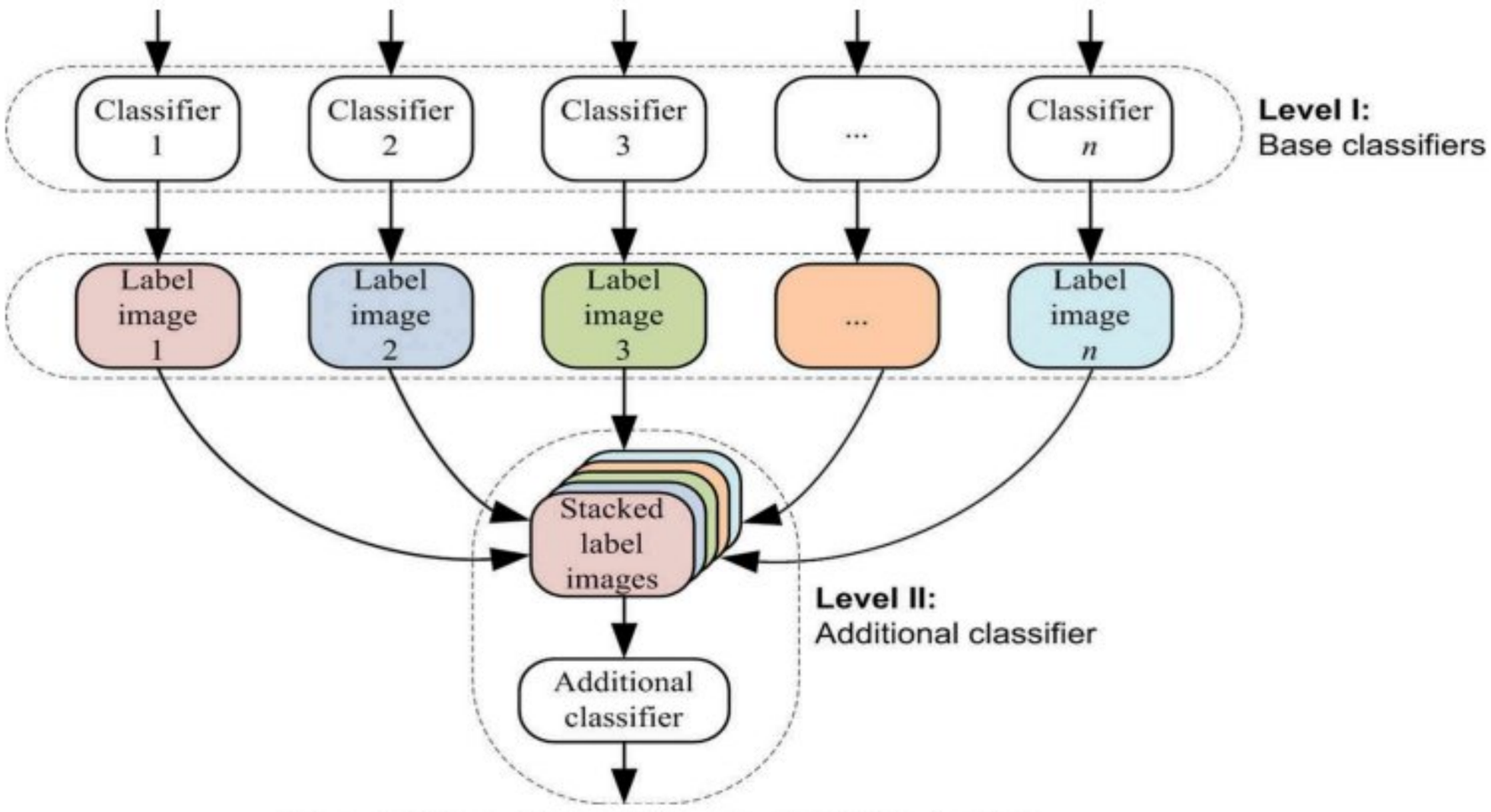
Crop mapping is an essential need in agriculture planning and management activities at national and global scales. A reliable crop map plays an important role with regard to various applications in environmental and agricultural management, such as crop inventory, crop insurance, yield estimation and the enforcement of quota limits Remote sensing and earth observation technologies exhibit valuable optical and radar sensors, offering various spatial and temporal resolutions. These sensors can be used for crop mapping and monitoring in a more efficient manner and with lower costs than traditional survey methods.

To date, optical satellites, such as Moderate Resolution Imaging Spectro radio-meter (MODIS), Landsat, National Oceanic and Atmospheric Administration (NOAA), Indian Remote Sensing (IRS), and Satellite Pour observation de la Terre (SPOT), have extensively been utilized in crop mapping studies at global, national, regional and local scales In recent years, have also been of interest to researchers for crop mapping and monitoring. Compared to the aforementioned optical sensors, these two satellites have a unique spectral band for analysing vegetation, known as the red edge (RE). Moreover, thanks to the simultaneous orbiting of five Rapid Eye sensors, it can frequently acquire images from a given area. These two advantages have distinguished Rapid Eye from the other optical satellite systems for agricultural applications

The primary observations of optical sensors for crop mapping are in the form of multispectral information in the range of visible to near-infrared (NIR) wavelengths, which characterize the reflectance behaviour of crop types However, vegetation indices (VIs), mainly based on red and NIR bands, are the most widely used products of optical sensors, providing more valuable information for discriminating various crop types. The normalized difference vegetation index (NDVI), simple ratio (SR) and enhanced vegetation index (EVI) are the most well-known VIs according to previous crop mapping studies Moreover, some applicable VIs based on RE channels, such as the red-edge Ndi red-edge SR and red-edge triangular vegetation index, were recently proposed and used in the literature In addition to the spectral bands and VIs, textural indicators have proven to be useful features for crop mapping. The most well known textural features already used for crop mapping are the parameters extracted from the grey level co-occurrence matrix (GLCM) of spectral channels. These features describe the spatial arrangement of grey levels in an image Compared to the optical sensors, fewer studies have so far employed synthetic aperture radar or full-polarimetric SAR sensors, such as the European Remote Sensing , the Advanced Synthetic Aperture Radar (ASAR) and Radar sat, for crop mapping on regional or local. The Unmanned Aerial Vehicle Synthetic Aperture Radar (UAVSAR) system, as an airborne Polisar sensor, has

been recently used in several diverse classification algorithms with a different nature and architecture have already been proposed for crop mapping. Some studies have used Bayesian networks or the maximum likelihood method These two traditional methods are parametric, i.e., they strongly depend on data distribution. Therefore, they may be limited in the classification of big data. Some other studies have used nonparametric methods, such as neural networks, support vector machines (SVM), or both. A neural network, which has a black box architecture, is known as an unstable classifier in the literature . By contrast, an SVM is known as an effiffiifficient and stable method for the classification of high-dimensional data in the literature. However, there are some major challenges for SVMs, such as the choice of optimum kernels and their parameters, and Other nonparametric algorithms used for crop mapping are tree-based classifiers, such as decision trees and random forest (RF) A decision tree is a weak classifier and may be faced with limitations in the classification of big data. Conversely, an RF, whose construction is based on a multitude of decision trees, has a high level of capability in this condition, which is comparable to an SVM. Moreover, an RF is preferred over an SVM thanks to its various advantages: an RF is much faster, more flexible and more robust in relation to the outliers, and has a much simpler structure. It has less concern for the choice of optimum parameters Another concept found in the literature for dealing with big data or high-dimensional feature space is stacked generalization or stacking. Stacking, as an effective decision fusion strategy, involves training a classifier to combine predictions obtained by several other classifiers Waska and Benedictus proposed an SVM-based stacking procedure to combine multitemporal SAR and optical imagery.

In other studies, SVM-based stacking has also been used for the classification of hyperspectral data Waska and van der Linden presented a multilevel object-based SVM procedure for classifying multitemporal SAR and optical data. Employing RF-based, as well as SVM-based, stacking for combining predictions at each level, their results indicated that RF-based stacking was able to offer greater accuracy than the SVM-based version. In all of the above studies, only labels of individual classifiers were used as feature inputs for stacking.



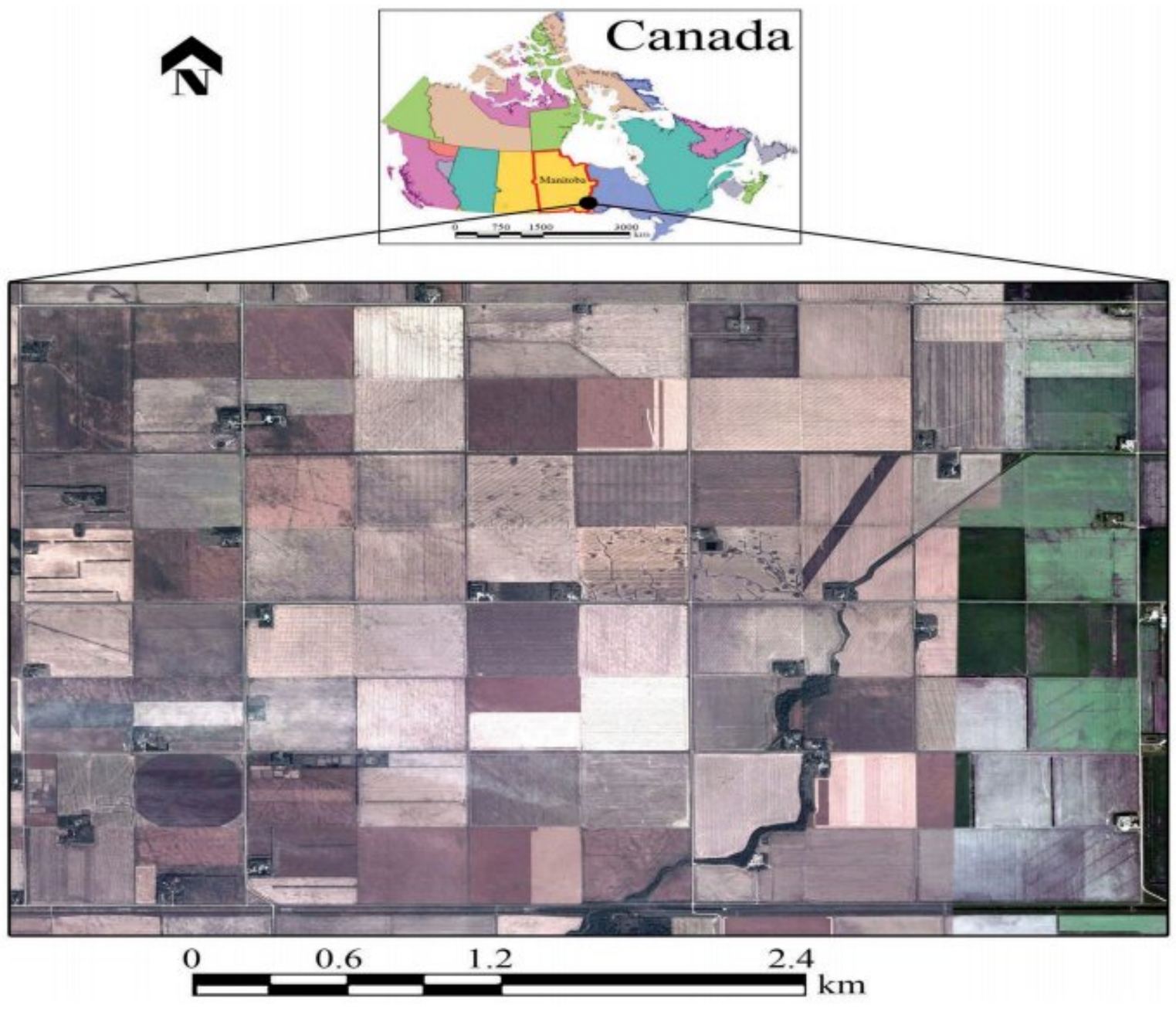
The outline of traditional stacking fusion.

This research work sought to establish an operational framework for crop mapping using bi-temporal of optical and SAR observations. In this context, we examined the capability and effectiveness of a diverse set of spectral, textural and polarimetric features in distinct feature subset groups for the classification of crop lands. The proposed methodology was based on the stacked generalization of an RF classifier. Unlike typical stacking methods, this method employed updated data rather than individual labels as new inputs for stacking. The proposed framework can be very beneficial for the operational crop mapping and monitoring from the freely available optical and radar earth observations, such as Sentinel-1and -2, Landsat-8 and the future Radar sat Constellation Mission (RCM).

EARTH OBSERVATIONS AND STUDY AREA

The study area of this paper was the southwest district of Winnipeg, Manitoba, Canada, which is covered by various annual crops (see Figure 2). This region is located from 47° 32‘ 16‘‘ to 48° 12‘ 56‘‘ N and from 97° 5‘ 2‘‘ to 97° 45‘ 13‘‘ W. The area of the region is near 400 ha. The mean elevation from sea level is about 784 ft. The data used in this paper were bi-temporal optical and radar images acquired by Rapid Eye satellites and the UAVSAR system. Rapid Eye is a spaceborne satellite, which has five spectral channels: blue (B), green (G), red (R), NIR and RE. In this paper, two optical images were collected on 5 and 14 July 2012. Both these images were orthorectified on the local North American 1983 datum (NAD-83) with a spatial resolution of about 5 m. The UAVSAR sensor is an airborne SAR sensor, which operates in the L-band frequency in full polarization mode (i.e., HH, HV, VH and VV). The radar images used in this paper were simultaneously acquired with the optical images. They were orthorectified on the World Geodetic System 1984 datum (WGS-84) with an SRTM3 digital elevation model. They were also multilocked by 2 pixels in azimuth and 3 pixels in range directions. Moreover, the despeckling process, using a 5 × 5 boxcar filter, was applied to the data in order to alleviate the speckle effect. The spatial resolution of these images was then approximately 15 m.

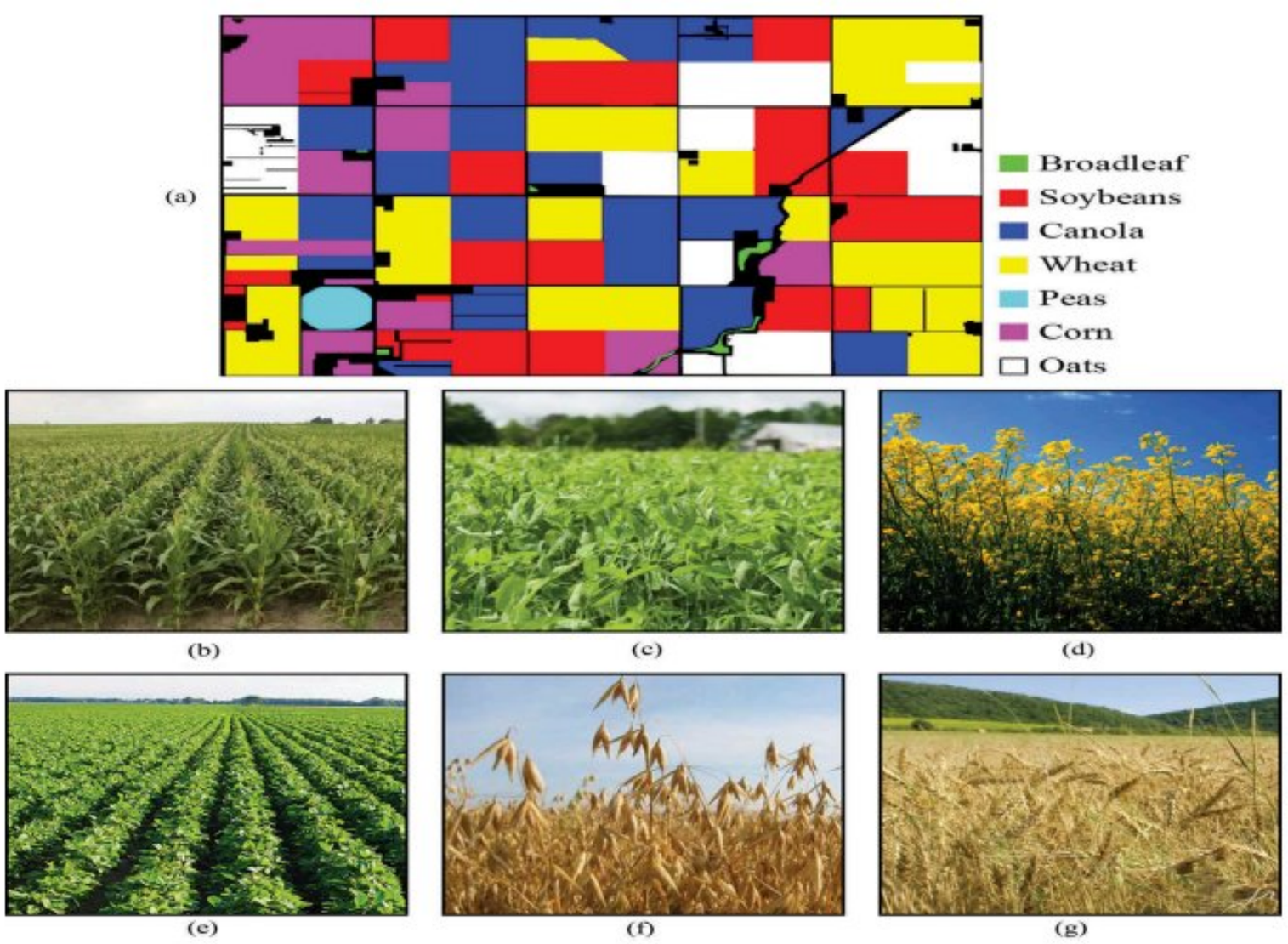
According to reference data for the study area, seven classes were found in this region as follows: canola, corn, soybeans, peas, oat, wheat and broadleaf. The number of samples for each class is presented here As can be seen, there is an imbalanced data distribution among the classes. Broadleaf and peas have fewer samples (i.e., minority classes), while other classes have much more samples (i.e., majority classes). In land-cover classification process, the collection of training and test samples is always a challenging and overwhelming task. Therefore, researchers are usually interested in obtaining the high accuracy with the inadequate data. For this purpose, we supposed that the training samples were limited for crop mapping in order to better assess the efficiency and capability of the traditional and proposed methods. In this paper, 5% of all samples, in two scenarios, were randomly stratified selected for the training purpose: (1) when there was a balanced data distribution among the classes and (2) when there was an imbalanced data distribution among the classes. These two types of sampling were accomplished in order to investigate the sensitivity of the methods to data distribution. The remaining samples were used for the accuracy assessment.



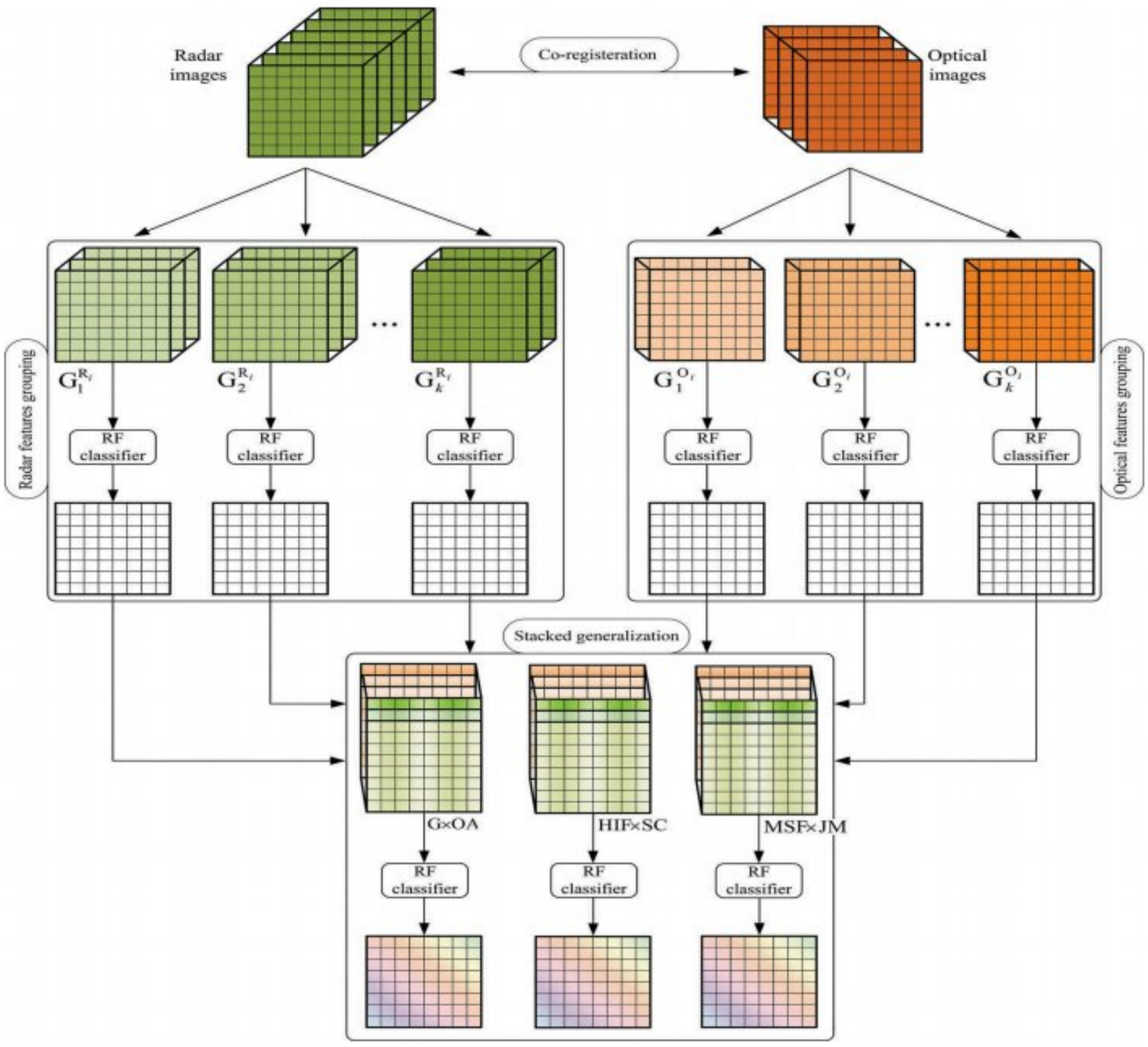
A subset of the study area, Winnipeg, Manitoba, Canada.

PROPOSED CROP MAPPING FRAMEWORK

The figure shown below illustrates the proposed crop mapping framework in this paper, based on a decision fusion strategy and the RF classifier from the bi-temporal UAVSAR and Rapid Eye images.



The first step, the bi-temporal Polisar and optical images were efficiently co-registered. The radar images had a coarser resolution; thus, the optical images were warped to be isidiametric with the radar images. Since the radar and optical images were geo-referenced at different datums and they had unequal pixel sizes, there were scale, rotation, and translation parameters for co-registering. For this purpose, the polynomial functions with degrees 1 (linear) and 2 (quadratic) were tested and the best results were then chosen based on root mean square error (RMSE). The experiments demonstrated that the quadratic polynomial (with RMSE = 0.36) was better than the linear polynomial (with RMSE = 1.55) for coregistering. In addition, nearest neighbour method was used for grey-level interpolation. Finally, all images had 15m spatial resolution.



The proposed RF-based framework for crop mapping.

Various polarimetric and optical features were then extracted from these images. One of the main goals of this research was to examine the effectiveness of these The proposed RF-based framework for crop mapping. 7228 I. KHOSRAVI AND S. K. ALAVIPANAH features for the classification of croplands in distinct feature subsets. For this purpose, these features were grouped. In this paper, we manually grouped the features based on their radar and optical natures and definitions. In fact, we put these features into several similar groups In the next step, each group was classified separately using a unique RF classifier. Finally, three novel stacking fusion strategies, based on an additional RF, were proposed. Unlike trade trigonal stacking methods updated data were used instead of unique predictionfor stacking. In this case, the updated data were built based on overall accuracy values, variable importance scores and Jeffffries-Matusita distances.

RANDOM FOREST ALGORITHM

For classifying each of the radar and optical groups, we used a standard RF classifier. Developed by Bierman the RF algorithm is an ensemble of decision trees. In fact, an RF combines a multitude of diverse decision trees to reduce the risk of oversighting. The RF trains these decision trees in parallel separately. This algorithm injects randomness into two steps of the training process. Firstly, training samples are divided into several subsamples by a random sampling method, i.e., a bootstrapping method. A subset of features from all features for each subsample is then chosen by a random subset feature selection method in this paper, 500 was considered for entrees. The root square of the number of features in each radar or optical group, as mentioned in the previous section, was considered for mTry, according to Bierman.

Features and formulas (G: groups, O: optical, i = 1 to 5)	
$G_1^{O_i}$	R_B: 440 – 510nm, R_G: 520 – 590nm, R_R: 630 – 685nm, R_{RE}: 690 – 730nm, R_{NIR}: 760 – 850nm R: Reflectance
$G_2^{O_i}$	$NDVI = (R_{NIR} - R_R)/(R_{NIR} + R_R)$
	$SR = R_{NIR}/R_R$
	$RGRI = R_G/R_R$
	$EVI = 2.5(R_{NIR} - R_R)/(R_{NIR} + 6R_R - 7.5R_B + 1)$
	$ARVI = (R_{NIR} - (2R_R - R_B))/(R_{NIR} + (2R_R - R_B))$
	$SAVI = (1 + 0.5)(R_{NIR} - R_R)/(R_{NIR} + R_R + 0.5)$
	$NDGI = (R_G - R_R)/(R_G + R_R)$
	$gNDVI = (R_{NIR} - R_G)/(R_{NIR} + R_G)$
	$MTVI2 = 1.5(1.2(R_{NIR} - R_G) - 2.5(R_R - R_G))/\sqrt{(2R_{NIR} + 1)^2 - (6R_{NIR} - 5\sqrt{R_R}) - 0.5}$
$G_3^{O_i}$	$NDVIre = (R_{NIR} - R_{RE})/(R_{NIR} + R_{RE})$
	$SRre = R_{NIR}/R_{RE}$
	$NDGIre = (R_G - R_{RE})/(R_G + R_{RE})$
	$RTVIcore = 100(R_{NIR} - R_{RE}) - 10(R_{NIR} - R_G)$
	$RNDVI = (R_{RE} - R_R)/(R_{RE} + R_R)$
	$TCARI = 3((R_{RE} - R_R) - 0.2(R_{RE} - R_G)(R_{RE}/R_R))$
	$TVI = 0.5(120(R_{RE} - R_G) - 200(R_R - R_G))$
	$PRI2 = R_{RE}/R_R$
$G_4^{O_i}$	μ_{PC1}, σ_{PC1}, HOM_{PC1}, CON_{PC1}, DIS_{PC1}, H_{PC1}, ASM_{PC1}, COR_{PC1} from GLCM of PC1
$G_5^{O_i}$	μ_{PC2}, σ_{PC2}, HOM_{PC2}, CON_{PC2}, DIS_{PC2}, H_{PC2}, ASM_{PC2}, COR_{PC2} from GLCM of PC2

In the prediction process for a test sample, the RF combines the outputs or labels of individual decision trees using a classical majority voting technique. This voting technique chooses class w_k. for sample p, when this class receives the highest number of votes among all votes obtained by all decision trees . Mathematically, class w_k. is chosen by the majority voting technique, if:

$$\sum_{i=1}^{L} d_{i,k} = \max_{j=1}^{c} \sum_{i=1}^{L} d_{i,j}$$

where $d_{i,j} \in \{0, 1\}$ is the decision value of the i th decision tree for class w_j {i = 1, 2, 3, . . ., L; j = 1, 2, 3, . . ., c}, and L and c are the number of decision trees and classes, respectively. Here, $d_{i,j}$ = 1, if the i th decision tree chooses class w_j , and 0 otherwise.

The RF algorithm does have a particular capability, the so-called 'variable importance', which is the ranking of features in a classification task. For determining the importance of a given feature, at first, the out-of-bag error (OOBE) is computed during the training process.

The OOBE is the mean prediction error on each training sample x_i, using only the decision trees that have been trained by the bootstrap sample in which xi is absent. After training, the values of that feature are then permuted among the training data, and the OOBE is re-computed using these perturbed data. The importance score for this feature is thus determined by averaging the difference of the OOBE before and after the permutation over all decision trees. The score can be normalized by the standard deviation of these differences.

Decision fusion strategies: stacking algorithm

Stacking decision fusion consists of training a classifier in order to combine the predictions of several other classifiers. First, all other algorithms are trained using the available data. Then, all the predictions of these algorithms, as new inputs, are used for an additional classification using a combiner algorithm to make a final prediction.In this paper, we proposed three stacking strategies to fuse the results of individual RF classifiers. The combiner in this case was an additional RF classifier. Unlike traditional stacking algorithms, we did not use the outputs of individual RFs as the inputs of the additional RF. Instead, we used an updated data for each strategy . The proposed strategies were as follows:

Strategy 1 (group × overall accuracy)

In Strategy 1, after the classification of each radar and optical features group, the data were updated as follows: each group was firstly multiplied by the overall accuracy obtained by the trained RF using that group. All the weighted groups were then stacked together (G × OA).

Strategy 2 (highest important features × importance score)

In this strategy, the data were updated as follows: in parallel to the classification of each group, the try number of highest important features (HIFk) was chosen, based on the variable importance property.

Strategy 3 (most separable features × separation value)

In Strategy 3, the data were updated as follows: before the classification of each group, the mTry number of the most separable features (MSFk) was chosen using an efficient separation measure. In this paper, the Jeffffries-Matusita (JM) distance, as an efficient and common measure of separation in the classification task, was employed. This distance can determine the separability of two classes for a given feature as follows :

$$JM = 2(1 - \exp(-B))$$

where B is the Bhattacharyya distance value deified as follows:

$$B = \frac{1}{8}(\mu_1 - \mu_2)^2 \frac{2}{\sigma_1^2 + \sigma_2^2} + \frac{1}{2}\ln\left(\frac{\sigma_1^2 + \sigma_2^2}{2\sigma_1\sigma_2}\right)$$

where μi and σi (i = 1,2) are the mean and variance for two classes and a feature. The JM value is in the range of [0, 2]. Zero indicates the weakest separation, while 2 indicates the highest separation among the classes for that feature (Zhang et al. 2016). After computing the JM-values, the most separable features of each group were multiplied by their mean JM-values over classes (JMi) and then stacked together (MSF × JM). The updated data was finally reclassifiified by an additional RF algorithm.

IMPLEMENTATION AND RESULTS

The training was carried out in two steps – Balanced and Imbalanced since the dataset was imbalanced as explained earlier. For the balanced training, 1000 samples were taken from each of the classes and were ordered randomly. This formed the training set. The next step was to follow the procedure mentioned earlier where the samples were divided into optical and radar feauture groups and each of the groups was then separately trained using an RF classifier. Initially, the results predicted by each of the groups was multiplied by the overall accuracy and then stacked for further training. The results obtained in this case are as shown below:

A Glance at Code used for Implementation

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score


Xtrain_group1_radar_date1 = balanced_training_data[:,[1, 2, 3, 4, 5,
6]]
classifier_radar1_date1 = RandomForestClassifier(n_estimators = 500,
max_features = 'sqrt')
classifier_radar1_date1.fit(Xtrain_group1_radar_date1, y_train)
X_test_radar1_date1 = data[:, [1, 2, 3, 4, 5, 6]]
y_test = data[:, 0]
y_predicted_radar1_date1 =
classifier_radar1_date1.predict(X_test_radar1_date1)
OA_group1_radar_date1 = accuracy_score(y_test,
y_predicted_radar1_date1)
print('Overall Accuracy of Radar Group 1 on date 1: %f\n'%
(OA_group1_radar_date1))


y_predict_train_group1_radar_date1 =
classifier_radar1_date1.predict(Xtrain_group1_radar_date1)
next_group1_date1_radar =
y_predict_train_group1_radar_date1*OA_group1_radar_date1
next_group1_date1_radar = np.reshape(next_group1_date1_radar, (-1,
1))


Xtrain_radar_date1 = np.hstack((next_group1_date1_radar,
```

```
next_group2_date1_radar, next_group3_date1_radar,
next_group4_date1_radar, next_group5_date1_radar,
next_group6_date1_radar, next_group7_date1_radar))


classifier_radar_date1 = RandomForestClassifier(n_estimators = 500,
max_features = 'sqrt')
classifier_radar_date1.fit(Xtrain_radar_date1, y_train)


X_test_radar_date1 =
np.hstack((y_predicted_radar1_date1*OA_group1_radar_date1,
y_predicted_radar2_date1*OA_group2_radar_date1,
y_predicted_radar3_date1*OA_group3_radar_date1,
y_predicted_radar4_date1*OA_group4_radar_date1,
y_predicted_radar5_date1*OA_group5_radar_date1,
y_predicted_radar6_date1*OA_group6_radar_date1,
y_predicted_radar7_date1*OA_group7_radar_date1))
X_test_radar_date1 = np.transpose(np.reshape(X_test_radar_date1, (7,
-1)))
y_predicted_radar_date1 =
classifier_radar_date1.predict(X_test_radar_date1)
OA_radar_date1 = accuracy_score(y_test, y_predicted_radar1_date1)
print('Overall Accuracy of radar data on date 1 : %f\n'%
(OA_radar_date1))
```

RADAR GROUPS FOR DATE 1	OVERALL ACCURACY
G1	60.33%
G2	52.10%
G3	51.28%
G4	49.59%
G5	61.88%

G6	50.09%
G7	50.47%

OPTICAL GROUPS FOR DATE 1	OVERALL ACCURACY
G1	66.67%
G2	59.53%
G3	65.67%
G4	45.32%
G5	38.95%

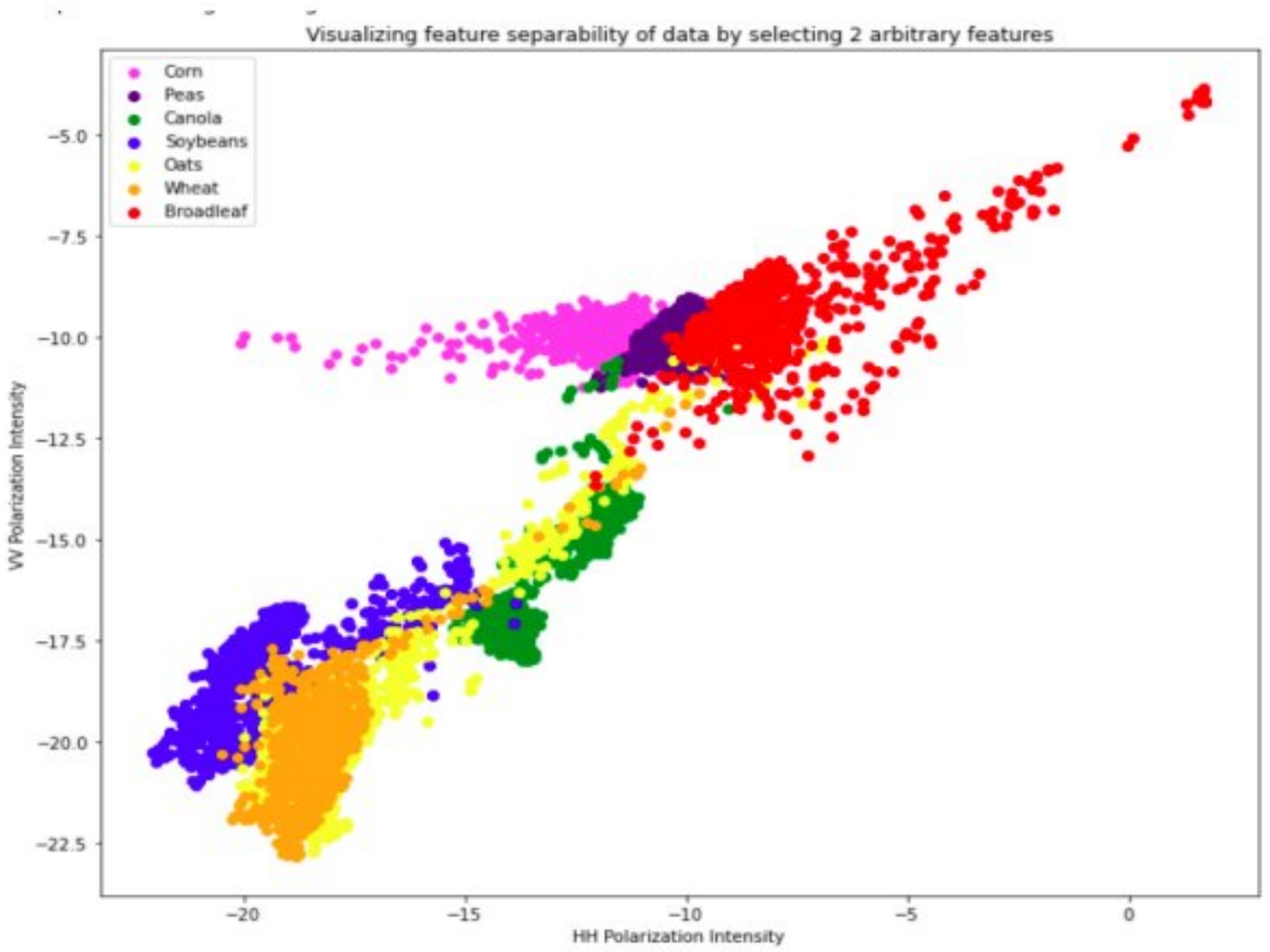
RADAR GROUPS FOR DATE 2	OVERALL ACCURACY
G1	75.24%
G2	48.05%
G3	45.68%
G4	37.48%
G5	75.96%
G6	63.58%
G7	66.97%

OPTICAL GROUPS FOR DATE 2	OVERALL ACCURACY
G1	66.28%
G2	40.33%
G3	56.25%

G4	37.55%
G5	32.05%

STACKED FEATURES	OVERALL ACCURACY
RADAR FEATURES - DATE 1	60.33%
RADAR FEATURES - DATE 2	75.24%
RADAR FEATURES	70.04%
OPTICAL FEATURES - DATE 1	66.67%
OPTICAL FEATURES - DATE 2	66.28%
OPTICAL FEATURES	59.33%
RADAR & OPTICAL FEATURES	64.03%

It was observed that the final overall accuracy was only 64.03% which is quite average for a seven class classification problem. In order to verify that the data indeed was separable, the feature separability of the data was plotted for two randomly selected features. The results are shown below and clearly indicate the separability of the data.



As an improvement, the features themselves were then multiplied with the overall accuracy value and then stacked together. The final stacked results are summarized in the table below:

STACKED FEATURES	OVERALL ACCURACY
RADAR FEATURES - DATE 1	60.42%
RADAR FEATURES - DATE 2	75.22%
RADAR FEATURES	86.87%
OPTICAL FEATURES - DATE 1	66.59%
OPTICAL FEATURES - DATE 2	65.86%
OPTICAL FEATURES	78.60%
RADAR & OPTICAL FEATURES	90.55%

Thus, the final accuracy can be observed to be over 90% which is a considerable improvement and compares well with research in similar fields.

Imbalanced Training

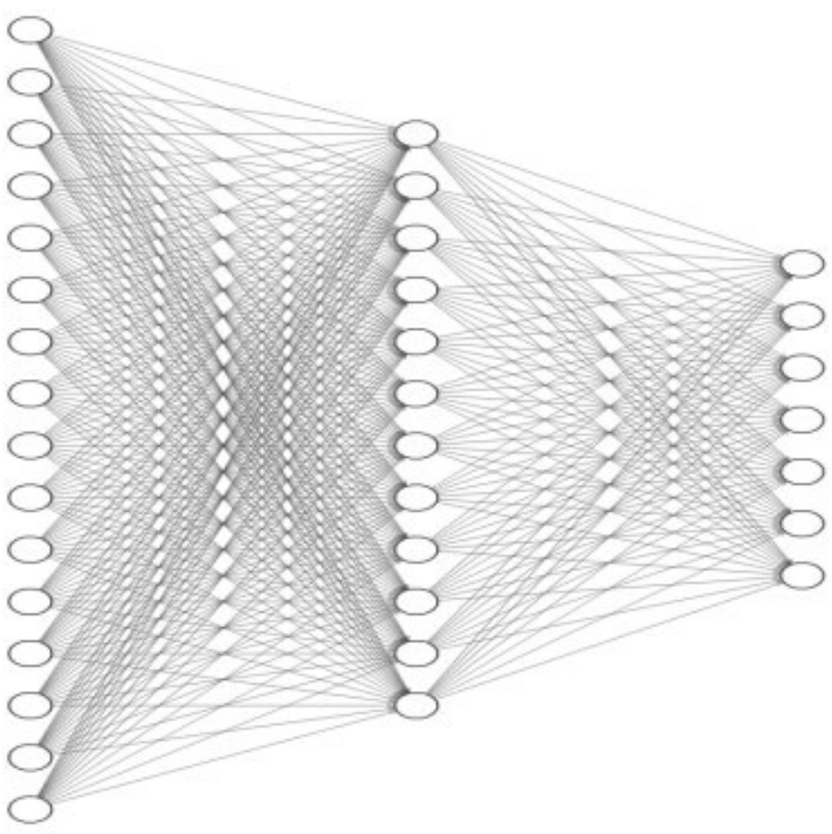
Random Forest Classifier was attempted for imbalanced training with two-thirds of the data as the training set and one-third as the test set. The classwise accuracies were rremarkably good at around 96% and thus the overall accuracy can be estimated to have come close to 100%. However, the time taken for training and memory requirement was too large and implementation was not possible due to resource constraints. Thus, it was resorted to use neural networks which can implement complex non-linear functions very efficiently and with small number of neurons. A neural network tries to model how the neurons in our brain work and maps it onto computer code. In a neuron, generally some processing happens in a single stage and then the information is passed on to the next neuron. Similarly, in a neural network, each layer of neurons process the input and pass on the result to the next layer of neurons. The processing stage involves a linear combination of inputs from previous neuron (Perceptron Learning) followed by an activation function to introduce non-linearities. There are several activation functions that can be used like sigmoid, ReLU, LeakyReLU, etc. The most commonly used one is the ReLU activation function which is what has been used in this case as well.

As we had earlier described, a neural network consists of several layers of neurons. The first layer is known as the input layer and the final layer is known as the output layer. All the layers in between are called hidden layers and these are the layers that implement the non-linearities in the hypothesis function. It is important to know how to choose the number of neurons in the input layer and the hidden layer as well as the number of hidden layers. These are known as hyperparameters and could affect the performance of the classifier,

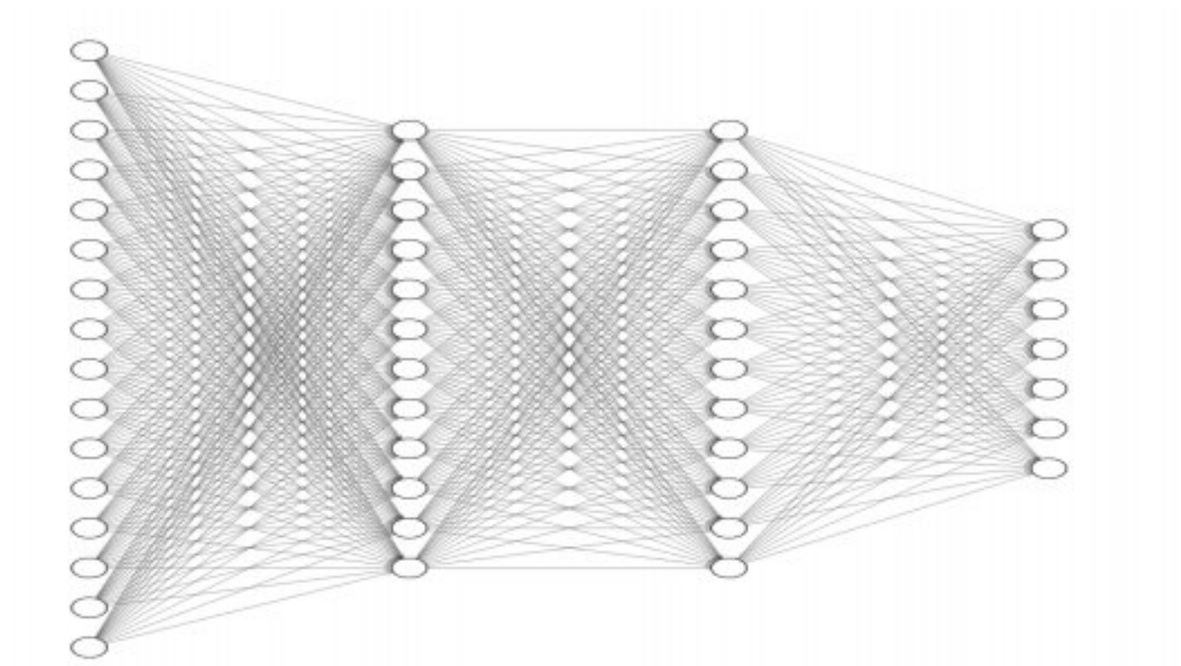
We now discuss how the architectures were arrived at. The number of neurons in the input layer is equal to the number of features i.e. 174 in this case. The number of neurons in the output layer was chosen to be equal to the number of classes with the intention that each neuron in the output layer would indicate the probability of the feature vector belonging to a particular class. A useful rule of thumb is to start with a single hidden layer and increase the number of hidden layers until the performance no longer improves. This was implemented and it was found that with an increase from one to two hidden layers, the performance actually decreased. This could have been due to poor choice of number of neurons in the two hidden layers but this was not tested further since the single hidden layer itself gave satisfactory results. A useful rule of thumb to choose the number of neurons is that the number of neurons is less than the number of neurons in the input layer and greater than the number of neurons in the output layer. In this case, the number of neurons was chosen to be the geometric mean of the number of neurons in the input and output layers.

After passing through the neural network, seven probability values are obtained and the predicted class is calculated by looking at the maximum of the probability values. The neural network architectures that were tried out and the corresponding results are as shown below:

1. Single Hidden Layer



2. Two Hidden Layers



3. Performance of Single Hidden Layer

```
Epoch 15/500
500/500 [==============================] - 2s 4ms/step - loss: 9.1349 - accuracy: 0.9198 - val_loss: 2.4975 - val_accuracy: 0.9570
Epoch 16/500
500/500 [==============================] - 2s 3ms/step - loss: 12.1884 - accuracy: 0.9125 - val_loss: 3.1065 - val_accuracy: 0.9492
Epoch 17/500
500/500 [==============================] - 2s 3ms/step - loss: 8.2772 - accuracy: 0.9260 - val_loss: 4.5093 - val_accuracy: 0.9531
Epoch 18/500
500/500 [==============================] - 2s 4ms/step - loss: 9.5611 - accuracy: 0.9198 - val_loss: 3.2605 - val_accuracy: 0.9336
Epoch 19/500
500/500 [==============================] - 2s 3ms/step - loss: 8.3618 - accuracy: 0.9232 - val_loss: 5.2934 - val_accuracy: 0.9336
Epoch 20/500
500/500 [==============================] - 2s 3ms/step - loss: 10.1683 - accuracy: 0.9126 - val_loss: 3.1727 - val_accuracy: 0.9570
Epoch 21/500
500/500 [==============================] - 2s 3ms/step - loss: 9.5007 - accuracy: 0.9239 - val_loss: 44.7430 - val_accuracy: 0.7812
Epoch 22/500
500/500 [==============================] - 2s 4ms/step - loss: 12.0754 - accuracy: 0.9141 - val_loss: 5.0989 - val_accuracy: 0.9180
Epoch 23/500
500/500 [==============================] - 2s 3ms/step - loss: 6.6663 - accuracy: 0.9294 - val_loss: 7.5614 - val_accuracy: 0.9062
Epoch 24/500
500/500 [==============================] - 2s 3ms/step - loss: 10.2480 - accuracy: 0.9125 - val_loss: 9.5240 - val_accuracy: 0.8750
Epoch 25/500
497/500 [============================>.] - ETA: 0s - loss: 7.2080 - accuracy: 0.9291INFO:tensorflow:Assets written to: /SingleLayerModel/assets
500/500 [==============================] - 2s 5ms/step - loss: 7.1794 - accuracy: 0.9293 - val_loss: 0.2448 - val_accuracy: 0.9922
Epoch 26/500
500/500 [==============================] - 2s 3ms/step - loss: 7.9251 - accuracy: 0.9266 - val_loss: 1.0116 - val_accuracy: 0.9688
Epoch 27/500
500/500 [==============================] - 1s 3ms/step - loss: 7.6420 - accuracy: 0.9257 - val_loss: 5.1829 - val_accuracy: 0.9648
Epoch 28/500
```

4. Performance of 2 Hidden Layers

```
Epoch 15/500
497/500 [============================>.] - ETA: 0s - loss: 55.1457 - accuracy: 0.7338INFO:tensorflow:Assets written to: /SingleLayerModel/assets
500/500 [==============================] - 3s 6ms/step - loss: 54.8758 - accuracy: 0.7337 - val_loss: 11.5499 - val_accuracy: 0.8828
Epoch 16/500
500/500 [==============================] - 2s 3ms/step - loss: 33.3535 - accuracy: 0.7633 - val_loss: 18.2218 - val_accuracy: 0.8008
Epoch 17/500
500/500 [==============================] - 2s 3ms/step - loss: 38.6675 - accuracy: 0.7456 - val_loss: 43.4058 - val_accuracy: 0.5547
Epoch 18/500
491/500 [============================>.] - ETA: 0s - loss: 27.7190 - accuracy: 0.7846INFO:tensorflow:Assets written to: /SingleLayerModel/assets
500/500 [==============================] - 3s 6ms/step - loss: 27.6354 - accuracy: 0.7842 - val_loss: 9.0864 - val_accuracy: 0.9062
Epoch 19/500
500/500 [==============================] - 2s 3ms/step - loss: 40.0771 - accuracy: 0.7650 - val_loss: 13.1498 - val_accuracy: 0.7734
Epoch 20/500
500/500 [==============================] - 2s 3ms/step - loss: 20.9534 - accuracy: 0.8002 - val_loss: 9.0233 - val_accuracy: 0.8633
Epoch 21/500
500/500 [==============================] - 2s 3ms/step - loss: 27.1688 - accuracy: 0.7877 - val_loss: 73.0210 - val_accuracy: 0.5586
Epoch 22/500
500/500 [==============================] - 2s 4ms/step - loss: 25.5658 - accuracy: 0.7926 - val_loss: 61.9695 - val_accuracy: 0.6172
Epoch 23/500
500/500 [==============================] - 2s 3ms/step - loss: 24.9595 - accuracy: 0.7956 - val_loss: 23.7484 - val_accuracy: 0.8086
Epoch 24/500
500/500 [==============================] - 2s 3ms/step - loss: 22.6854 - accuracy: 0.8032 - val_loss: 10.8737 - val_accuracy: 0.8828
Epoch 25/500
500/500 [==============================] - 2s 4ms/step - loss: 21.9678 - accuracy: 0.8109 - val_loss: 9.3023 - val_accuracy: 0.8555
Epoch 26/500
500/500 [==============================] - 2s 3ms/step - loss: 26.8509 - accuracy: 0.7889 - val_loss: 18.1688 - val_accuracy: 0.7461
Epoch 27/500
500/500 [==============================] - 2s 3ms/step - loss: 23.8704 - accuracy: 0.7999 - val_loss: 10.0237 - val_accuracy: 0.8789
Epoch 28/500
```

5. Predictions on Test Samples for Single Hidden Layer Case

```
Probability Values generated by the model :

[[0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 1.0000000e+00
  3.8196823e-24 0.0000000e+00]
 [0.0000000e+00 0.0000000e+00 2.8908889e-34 0.0000000e+00 9.9998927e-01
  1.0762273e-05 3.9544315e-37]
 [0.0000000e+00 0.0000000e+00 0.0000000e+00 1.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00]
 [0.0000000e+00 0.0000000e+00 1.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00]
 [0.0000000e+00 0.0000000e+00 0.0000000e+00 1.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00]
 [0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  1.0000000e+00 0.0000000e+00]
 [1.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00]
 [0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  1.0000000e+00 0.0000000e+00]
 [0.0000000e+00 0.0000000e+00 0.0000000e+00 1.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00]]

Class Selected based on Maximum Probability :

[5 5 4 3 4 6 1 1 6 4]

Actual Class :

[5 6 4 3 4 6 1 1 6 4]
```

A Glance at Code Used For Implementation

```
#Creating the Architecture for the Neural Network with a single hidden layer

model = keras.Sequential([

keras.layers.Dense(units = 35, activation = 'relu'),
keras.layers.Dense(units = 7, activation = 'softmax')
])

model.compile(optimizer = 'adam', loss =
tf.losses.CategoricalCrossentropy(from_logits=True), metrics = ['accuracy'])


#Training the Created Neural Network with the Training Set

callbacks = [
tf.keras.callbacks.ModelCheckpoint(filepath = '/SingleLayerModel', save_best_only =
True, monitor = 'val_accuracy'),
tf.keras.callbacks.EarlyStopping(monitor = 'val_accuracy', patience = 20)
]
model.fit(
train_dataset.repeat(),
epochs = 500,
steps_per_epoch = 500,
validation_data = test_dataset.repeat(),
validation_steps = 2,
callbacks = callbacks
)


#Creating the architecture for the neural network with two hidden layers


model = keras.Sequential([
keras.layers.Dense(units = 35, activation = 'relu'),
keras.layers.Dense(units = 35, activation = 'relu'),
```

```
keras.layers.Dense(units = 7, activation = 'softmax')
])
model.compile(optimizer = 'adam', loss =
tf.losses.CategoricalCrossentropy(from_logits=True), metrics = ['accuracy'])
```

CONFUSION MATRICES

Since the dataset is imbalanced, it was estimated that overall accuracy might not indicate an actual good performance by the algorithms. Thus, the classical confusion matrix for the imbalanced dataset along with calculations for precision, recall and F1-scores was computed for the cases of both balanced and imbalanced training. The results are as shown below:

1. **Balanced Training Case**

		PREDICTED CLASS							
		CLASS 1	CLASS 2	CLASS 3	CLASS 4	CLASS 5	CLASS 6	CLASS 7	RECALL
ACTUAL CLASS	CLASS 1	37143	28	451	130	639	90	680	94.85%
	CLASS 2	0	3468	117	2	10	0	0	96.41%
	CLASS 3	7	163	75187	88	73	94	60	99.36%
	CLASS 4	298	22	5349	59827	549	8020	1	80.78%
	CLASS 5	137	0	164	57	40428	6274	56	85.81%
	CLASS 6	424	12	183	387	6416	77596	55	91.21%
	CLASS 7	0	0	0	0	0	0	1143	100%
	PRECISION	97.72%	93.91%	91.63%	98.90%	84.02%	84.28%	57.32%	

F1-Scores -->

CLASS 1	96.26%
CLASS 2	95.14%
CLASS 3	95.34%
CLASS 4	88.93%
CLASS 5	84.91%
CLASS 6	87.61%
CLASS 7	72.87%
MACRO AVG	88.72%

For the balanced training case, we observe that the results are quite satisfactory except for class 7 which has a high recall but a low precision. The harmonic mean of the precision and recall gives the F1-Score for each of the classes and a higher percentage gives better performance. The macro average of the F1-Score which is in this case found out to be 88.72% is an indicator of the performance of the algorithm. The measure obtained is relative and research work in similar fields needs to be observed and compared to see if the classifier actually performs well. However, qualitatively, we can observe that an F1-Score of close to 90% appears to be quite satisfactory and tells us about the correctness of the algorithm. There is also a term known as the micro average F1 Score which is synonymous with the overall accuracy metric that we had earlier calculated. Another interesting observation that we can make from the confusion matrix is the fact that classes 5 and 6 are not easily separable and this tells us that the crops corresponding to these two classes are actually quite similar with respect to their radar and optical features and thus, some other features are necessary to distinguish them. This observation was made on the basis of the fact that there was a significant number of misclassifications of class 4 as class 6 and vice-versa. Thus, it is possible to make such interesting observations from the confusion matrix which would not have been possible in the case of a simple overall accuracy metric.

2. **Imbalanced Training**

ACTUAL \ PREDICTED CLASS	CLASS 1	CLASS 2	CLASS 3	CLASS 4	CLASS 5	CLASS 6	CLASS 7	RECALL
CLASS 1	38639	30	79	328	8	24	53	98.67%
CLASS 2	27	3532	12	23	0	3	0	98.19%
CLASS 3	38	48	75246	218	87	33	2	99.44%
CLASS 4	486	132	124	72951	129	239	5	98.49%
CLASS 5	1482	4	85	355	42553	2486	150	90.32%
CLASS 6	241	0	107	281	3803	80539	102	94.67%
CLASS 7	235	2	18	4	5	25	854	74.72%
PRECISION	93.90%	94.24%	99.44%	98.37%	91.34%	96.63%	73.24%	

F1-Scores -->

CLASS 1	96.23%
CLASS 2	96.17%
CLASS 3	99.44%
CLASS 4	98.43%
CLASS 5	90.83%
CLASS 6	95.64%
CLASS 7	73.97%
MACRO AVG	92.96%

For the case of imbalanced training, we can observe that the individual class F1-Scores have in general higher values as compared to balanced training. This tells us that imbalanced training gives rise to a better redistribution between precision and recall thus giving rise to better F1-Scores (This arises from the no free lunch theorem according to which if we try to improve either precision or recall the other quantity faces a deterioration and thus there is a tradeoff). This redistribution can be observed very significantly for class 7 where precision was improved by almost 20% by sacrificing the 100% recall. Thus, we arrive at a conclusion that imbalanced training is better as compared to balanced training for this particular dataset. We however also point out that only 1000 samples were used for balanced training while two-thirds of the dataset was used for imbalanced training which could have been a contributing factor as well. This was not explored very much in detail owing to the time constraints.

CONCLUSION

Thus, we have finally obtained an overall accuracy of over 90% for both the balanced and imbalanced cases with higher overall accuracy for the imbalanced case. Thus, we can regard the training to be successful but also think of improving the overall accuracy as crop mapping is a critical application and even a rare mistake could cause severe economic losses if the algorithm is put to use. The F1-score values further validate our inferences.

FUTURE WORK

- More advanced methods can be explored for stacking fusion strategies - Variable feature selection, Bhattacharya's distance based selection, etc.
- More advanced and standard architectures for training the neural networks could offer improved accuracy
- Study on how data was collected and how the radar and optical features were combined - Processes like coregistration and orthorectification

REFERENCES

- Iman Khosravi & Seyed Kazem Alavipanah (2019) A random forest-based framework for crop mapping using temporal, spectral, textural and polarimetric observations, International Journal of Remote Sensing, 40:18, 7221-7251, DOI: 10.1080/01431161.2019.1601285
- https://www.analyticsvidhya.com/blog/2021/04/getting-into-random-forest-algorithms/